TÉCNICAS DE PROGRAMACIÓN

ESTRUCTURAS DE DATOS BÁSICAS

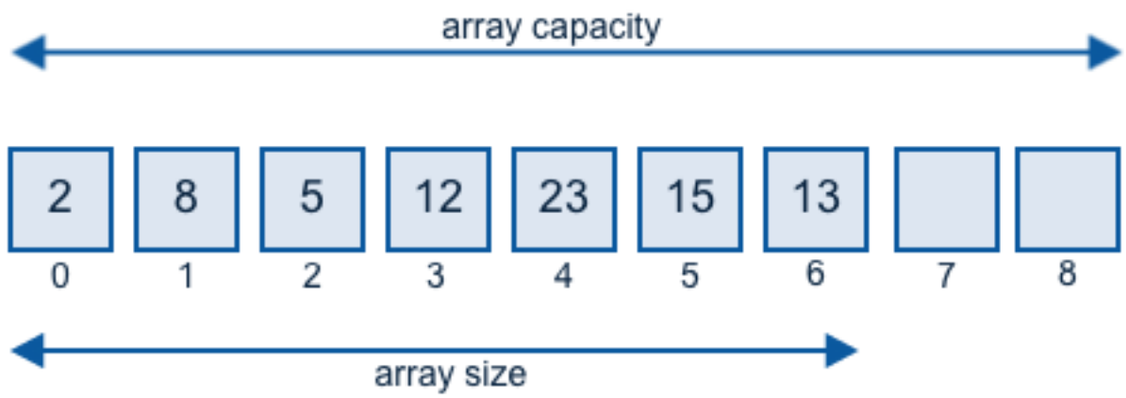
UNA ESTRUCTURA DE DATOS ES UNA FORMA PARTICULAR DE ORGANIZAR DATOS EN UNA COMPUTADORA PARA QUE PUEDA SER UTILIZADO DE MANERA EFICIENTE.

DIFERENTES TIPOS DE ESTRUCTURAS DE DATOS SON ADECUADOS PARA DIFERENTES TIPOS DE APLICACIONES, Y ALGUNOS SON ALTAMENTE ESPECIALIZADOS PARA TAREAS ESPECÍFICAS.

ARRAYS (ARREGLOS)

ARRAYS

https://docs.python.org/3.6/library/array.html



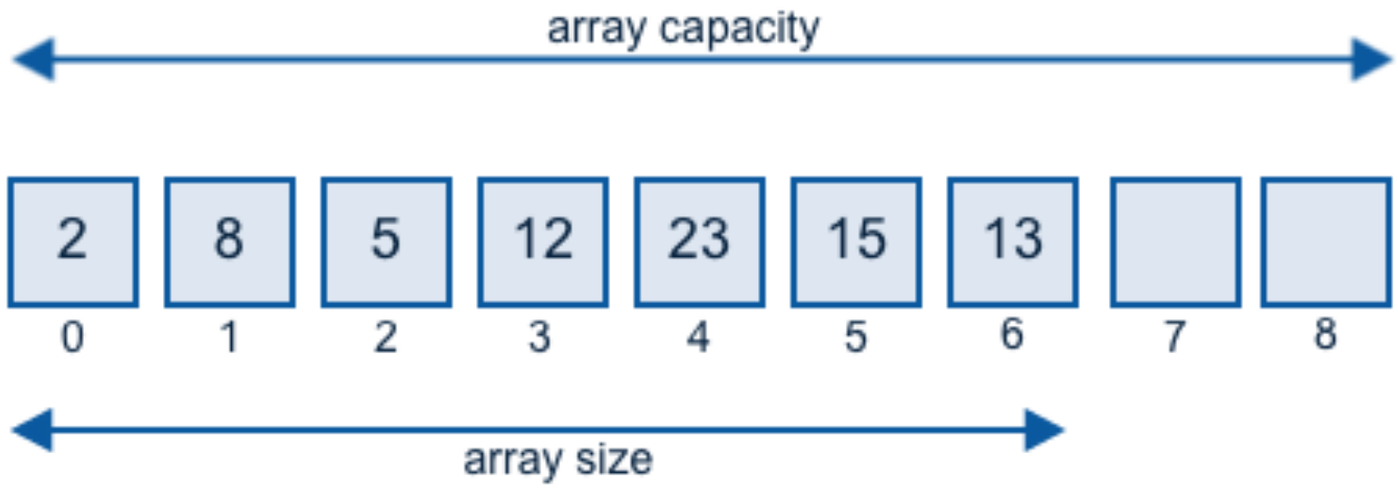
- ES LA ESTRUCTURA MAS BÁSICA DE DATOS ORDENADA QUE EXISTE
- TAN BÁSICA QUE ACTUALMENTE SOLO SE USA EN CASOS ESPECIALES
- POR DEFINICIÓN SOLO DEBE PERMITIR UN ÚNICO TIPO DE DATO
- POR DEFINICIÓN DEBE DE SER DE TAMAÑO DEFINIDO Y FIJO
- DESDE EL PUNTO DE VISTA DE MEMORIA SON MUY EFICIENTES

Caso de uso: https://www.python.org/doc/essays/list2str/

LISTS (LISTAS)

LISTAS

https://docs.python.org/3.6/tutorial/datastructures.html



- BÁSICAMENTE ES LO MISMO DE UN ARREGLO
- ES UNA ESTRUCTURA ORDENADA
- ES UN ELEMENTO MUTABLE: PUEDE CAMBIAR DE DIMENSIÓN
- PUEDE CONTENER MÚLTIPLES TIPOS DE ELEMENTO
- ESTA OPTIMIZADA PARA MANIPULACIÓN (LECTURA, ESCRITURA)
- OCUPA MÁS ESPACIO EN MEMORIA A COMPARACIÓN DE UN ARREGLO

LISTAS/ MÉTODOS Y OPERACIONES

Metodo	Descripción
list.append(x)	Agrega un elemento x al final de la lista
list.extend(iterable)	Agrega todos los elementos en iterable al final de la lista
list.insert(i, x)	Inserta un elemento x en el indice i
list.remove(x)	Elimina el cemento x
list.pop(i)	Remueve y regresa el elmentento en la posición i, si no existe i, se elimina y regresa el ultimo elemento de la lista
list.clear()	Elimina todos los elementos en la lista
list.index(x[, start[, end]])	Regresa el indice que corresponde al elemento x en la lista, si este elemento no existe se lanza un error
list.count(x)	Regresa el número de veces que elemento x, aparece en la lista
list.reverse()*	Le da la vuelta al la lista
list.copy()*	Regresa una nueva instancia de la lista, con los mismo elementos que la original

*


MARTY, DEBES VOLVEL AL PASADO
A ENSEÑAR MUTABILIDAD Y ESPACIOS DE MEMORIA

ELEMENTOS UNICOS

Crean una función que reciba una lista de números, y regrese un arreglo que no contenga elementos repetidos

Asume que siempre el argumento es un arreglo de números enteros de dimensión 0 a infinito

Entrada: [3, 57, 24, -37, 3, 17, 5, 5, 57]

Salida: [3, 57, 24, -37, 17, 5]

Entrada: [5, 5, 5, 5, 5, 5]

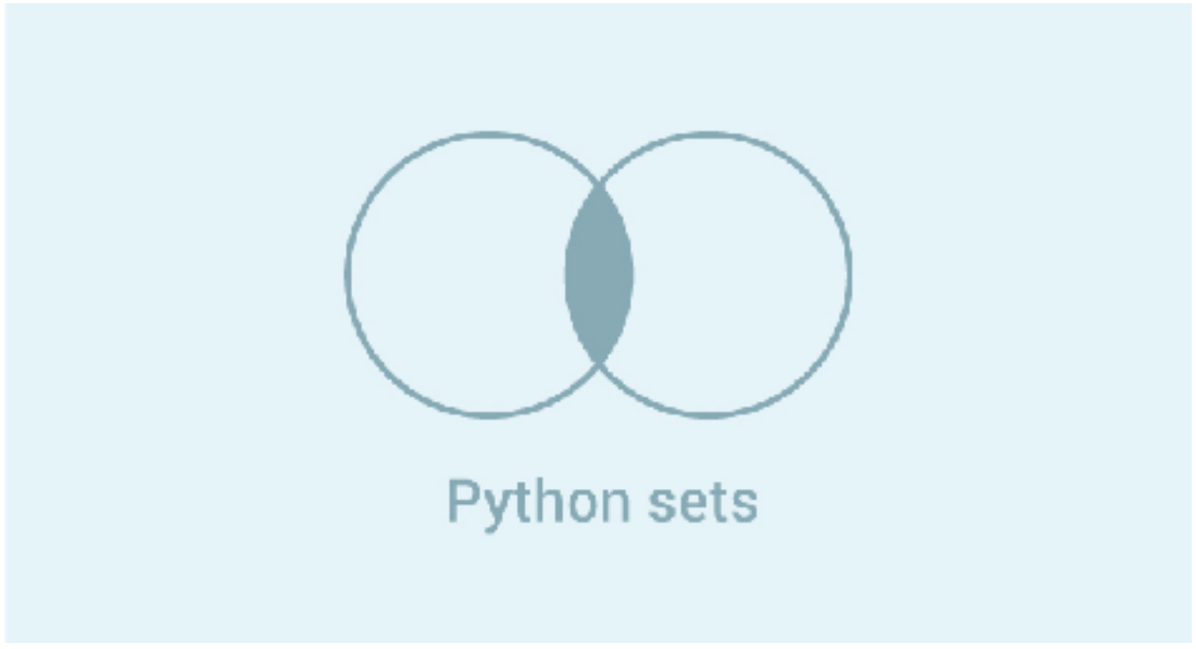
Salida: [5]

limpia_duplicados

SETS

SETS

https://docs.python.org/3.6/library/stdtypes.html#set



- CUENTA CON LA MISMA ESTRUCTURA EN MEMORIA QUE UNA LISTA
- ES UNA ESTRUCTURA NO ORDENADA
- SE CARACTERIZA POR QUE NO PUEDE CONTENER ELEMENTOS IGUALES
- PUEDE CONTENER ELEMENTOS DE MULTIPLES TIPOS*
- IDEAL PARA DETERMINAR UNIONES, INTERSECCIONES, DIFERENCIAS DE INFORMACIÓN

sets

SETS/ MÉTODOS Y OPERACIONES

Metodo	Descripción
set.isdisjoint(otro)	Regresa True si otro no tiene elementos en común
set.issubset(otro) set <= otro	Regresa True si todos los elementos de otro están en set
set.issuperset(otro) set >= otro	Regresa True si todos los elementos de set están en otro
set.union(otro) set | otro | …	Regresa un nuevo set con la union de otro y set
set.intersection(otro) set & otro & …	Regresa un nuevo set con la intersección entre otro y set
set.difference(otro) set - otro - …	Regresa un nuevo set con la diferencia entre otro y set
set.symmetric_difference(otro) set ^ otro ^ …	Regresa un nuevo set los elementos que no coincidan entre otro y set
set.copy()	Regresa una nueva instancia de la lista, no los mismo elementos que la original
set.add(x)	Agrega x al set
set.remove(x)	Remueve x del set. Lanza un error si el elemento no existe
set.discard(x)	Si x esta prensete en set, remueve el elemento
set.pop()	Remueve y regresa un elemento arbitrario en set. Lanza un error si no existen más elementos.
set.clear()	Remueve todos los elementos

SETS

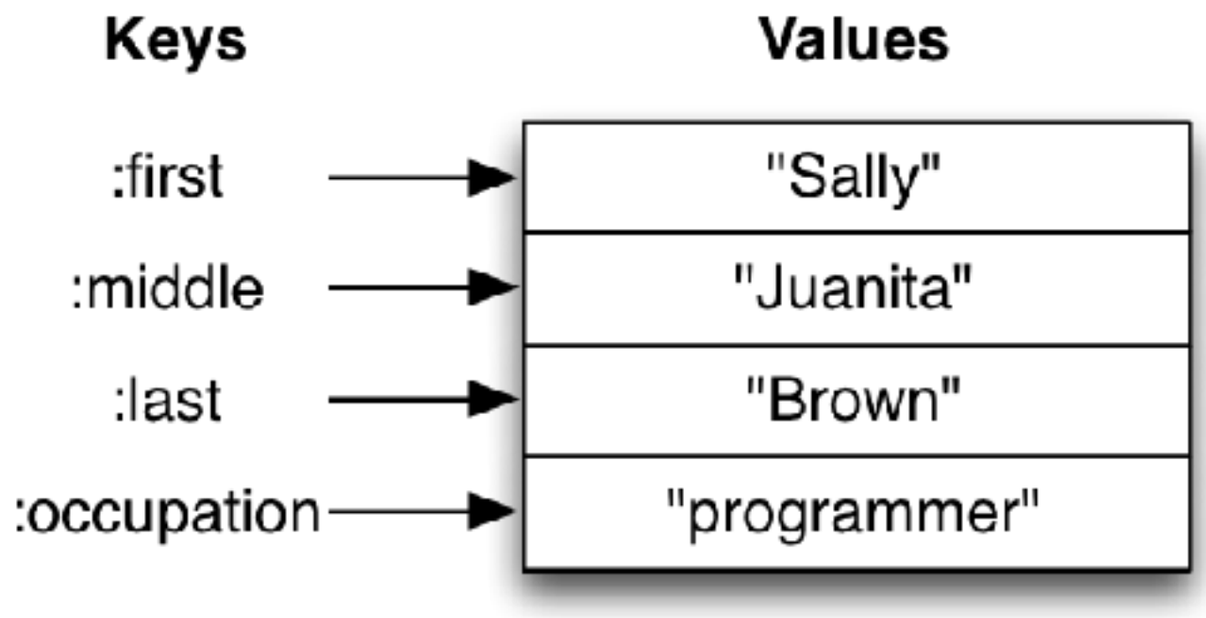
```
frozen = frozenset([1, 2, 3, 3, 3])
print(frozen) # frozenset({1, 2, 3})

# frozen.add(5) # Error
# frozen.discard(3) # Error
# frozen.remove(3) # Error
```

DICTIONNAIRES

(DICCIONARIOS)

DICTIONARIES

https://docs.python.org/3.6/library/stdtypes.html#dict



- ES UNA ESTRUCTURA DE DATOS NO ORDENADA BASADA EN KEYS (LLAVES)
- ES UNA ESTRUCTURA NO ORDENADA
- NOS PERMITE ALMACENAR CUALQUIER TIPO DE DATO ASOCIÁNDOLO A UN KEY
- UNA UNA ESTRUCTURA POPULARIZADA EN LOS ÚLTIMOS AÑOS POR EL FORMATO JSON

Metodo	Descripción
dict.keys()	Regresa una lista con todas las llaves disponibles
dict.values()	Regresa una lista con todos los valores asociados
dict.items()	Regresa una lista de tuplas, con los keys y sus valor asociado
dict.get(key, default)	Si existe el elemento key retorna el valor asociado, de lo contrario retorna el valor default
dict.pop(key, default)	Si existe el elemento key retorna el valor asociado y lo elimina de memoria, de lo contrario retorna el valor default
dict.update(other)	Actualiza el dict con los valores en other
dict.clear()	Elimina todos los elementos en el diccionario
dict.copy()	Regresa una nueva instancia del diccionario, con los mismo elementos del original

OBTÉN FACTORES

Completa la función *factors_range*, la cual recibe *n* y *m*, un rango de numero enteros positivos y regresa un diccionario que contiene los factores de cada número en el rango.

- **Para cada elemento en el rango se debe de regresar un arreglo con sus factores.**
- **El valor key de cada element en el diccionario es el mismo número**
- **En caso de que el elemento sea primo, regresar None1**

Entrada: 1, 5

Salida: {1: None, 2: None, 3: None, 4: [2], 5: None}

Entrada: 22, 25

Salida: {22: [2, 11], 23: None, 24: [2, 3, 4, 6, 8, 12], 25: [5]}

Factores

SWICHERS

```
switcher = {
        0: "zero",
        1: "one",
        2: "two",
    }

print(switcher.get(0, "nothing"))
print(switcher.get(5, "nothing"))
```

swichers

1. ARREGLOS
2. LISTAS Y SUS FUNCIONES BÁSICAS
3. SETS Y SUS FUNCIONES BÁSICAS
4. DICCIONARIOS Y SUS FUNCIONES BÁSICAS
5. SWICHERS